Aggressive Quadrotor Flight Using Dense Visual-Inertial Fusion

Yonggen Ling, Tianbo Liu, and Shaojie Shen

***Abstract*— In this work, we address the problem of aggressive flight of a quadrotor aerial vehicle using cameras and IMUs as the only sensing modalities. We present a fully integrated quadrotor system and demonstrate through online experiment the capability of autonomous flight with linear velocities up to 4.2 m/s, linear accelerations up to 9.6 m/s^2, and angular velocities up to 245.1 degree/s. Central to our approach is a dense visual-inertial state estimator for reliable tracking of aggressive motions. An uncertainty-aware direct dense visual tracking module provides camera pose tracking that takes inverse depth uncertainty into account and is resistant to motion blur. Measurements from IMU pre-integration and multi-constrained dense visual tracking are fused probabilistically using a semi-tightly-coupled, optimization-based sensor fusion framework. Extensive statistical analysis and comparison are presented to verify the performance of the proposed approach. We also release our code as open-source ROS packages.**

I. Introduction

Autonomous micro aerial vehicles (MAVs) have low-cost and superior mobility advantages, making them ideal robotic platforms for a wide range of applications such as aerial photography, surveillance, and search and rescue. These aerial robots are able to, in principle, navigate quickly through 3-D unstructured environments and provide fast response in hazardous environments that are dangerous or inaccessible for humans. The ability to achieve high-speed and aggressive flight is essential in such time-critical missions. However, there are still significant research and engineering challenges due to the lack of GPS measurements and potential sensor failures during fast motions. In particular, reliable state estimation using lightweight, off-the-shelf sensors is still the foremost important component for aggressive autonomous flight. Inspired by our earlier results towards vision-based high-speed quadrotor flight [1], in this work, we take another step forward using advanced methods in dense visual tracking and optimization-based sensor fusion to achieve robust state estimation. We show that, with only off-the-shelf cameras and IMUs as sensing modalities, we are able to achieve autonomous flight with linear velocities up to 4.2 m/s, linear accelerations up to 9.6 m/s^2, and angular velocities up to 245.1 degree/s.

As demonstrated in a wide body of literature, cameras are the ideal sensor for tracking of slow to moderate motions using feature-based methods. However, motion blur caused by aggressive motions can seriously downgrade feature detection and tracking performance. Recent advances in direct dense tracking have shown good adaptability to motion blur or textureless environments [2]–[4]. These methods directly operate on image intensities and make full use of all the available information within an image. Bypassing the feature processing pipeline eliminates some of the issues in feature-based methods. However, even dense methods are subject to failure during aggressive motions as images can be severely blurred. IMUs generate noisy but outlier-free measurements, making them great for tracking of short-term fast motions. However, low-cost MEMS IMUs suffer significant drift in the long term. We believe that combining the complementary natures of dense visual tracking and IMU measurements opens up the possibility of reliable tracking of aggressive motions.

All authors are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. {ylingaa, tliuam}@connect.ust.hk, eeshaojie@ust.hk

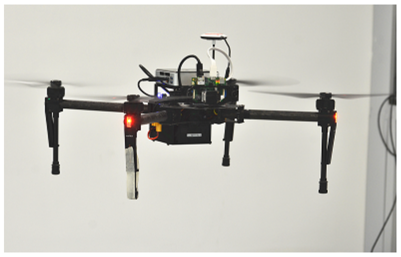


Fig. 1. Our experimental platform is the DJI M100 quadrotor equipped with an onboard computer (Intel i5-4250U 1.3GHz dual-core CPU) and a VI-Sensor (forward-facing stereo cameras and a MEMS IMU). The platform weights 3210g in total.

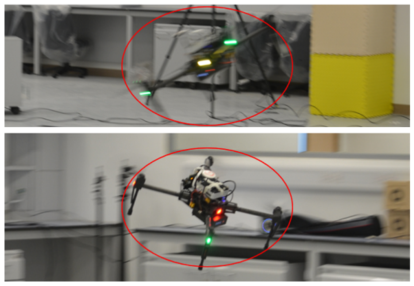


Fig. 2. Snapshots during aggressive autonomous flight experiment. We highlight the position of the robot with red circles. The maximum linear velocity, linear acceleration and angular velocity are 4.2 m/s, 9.6 m/s^2 and 245.1 degree/s respectively.

The key contribution of this work is a robust and fully integrated real-time solution for aggressive quadrotor flight. Our method uses the information from a pair of calibrated stereo cameras and a MEMS IMU and runs onboard a moderate computer (Fig. 1). The focus of this work is a semi-tightly-coupled, probabilistic, optimization-based estimation method that fuses pre-integrated IMU measurements and multi-constrained relative pose measurements from a depth uncertainty-aware dense visual tracking module. Our estimator actively searches for multi-constrained dense alignments between frames within a sliding window. This loop-closure-like method enables the estimator to recover from complete loss of visual tracking and eliminate drift after very aggressive motions. In addition, we initialize the incremental rotation for dense tracking using the angular prior from IMU measurements, which greatly improves the convergence property during aggressive motions. We release our code as open-source ROS packages with relevant video demonstrations. There are available at: `https://github.com/ygling2008/dense_new`.

The estimator of our proposed system is an extension of our preliminary work [4], with improvements in uncertainty-aware dense tracking and robust graph-based optimization. We believe that we are the first to introduce a practical visual-inertial system for aggressive autonomous flight of quadrotors.

The rest of the paper is structured as follows. In Sect. II, we review the state-of-the-art scholarly work. An overview of the system is presented in Sect. III. Details of the system are discussed in Sect. IV. Implementation details and experimental evaluations are presented in Sect. V. Sect VI draws the conclusions and points out possible future extensions.

II. Related Work

There has been extensive scholarly work in visual-inertial state estimation. Visual measurements can be calculated from different camera configurations, such as monocular [1, 5]–[8], stereo [9], or RGB-D cameras [10]. The majority of these approaches rely on detecting and tracking of sparse features across multiple frames. Though feature-based methods are well-developed, they depend heavily on image quality and are subject to failure when cameras undergo aggressive motions that lead to severe motion blur. With recent advances in high-performance mobile computing, direct dense methods have become popular [2, 3, 11]–[13]. By directly minimizing the photometric intensity error between images, these methods eliminate the feature processing pipeline, making them more resistant to image blur, provided that two images are about equally blurred.

It is straightforward to apply some variations of Kalman filtering [1, 7, 10] to loosely fuse visual and inertial measurements. The high-level effect of such fusion is the smoothing of vision-based tracking, and the use of IMUs for short-term motion prediction when visual tracking fails. In loosely-coupled methods, visual measurements are usually presented in the form of relative pose transformations, while leaving the visual pose tracking as a black box. This leads to lower computational complexity at the cost of suboptimal results. Recent developments in visual-inertial fusion indicate that tightly-coupled methods outperform their loosely-coupled counterparts in terms of estimation accuracy [5, 6, 8, 9, 14]. But this comes at the cost of higher computational complexity.

While dense methods are well-established in vision-only settings using RGB-D [11, 13], stereo [15] and monocular [3, 12] cameras, few visual-inertial fusion approaches incorporates dense tracking. Most similar to our work is [16], where dense tracking results are loosely fused with inertial measurements using an extended Kalman filter. No multi-constrained measurements or graph-based optimization are used in [16], thus limiting success in tracking failure recovery. We show through online experiment that our approach outperforms that in [16] during aggressive motions.

III. System Overview

The pipeline of our proposed system is illustrated in Fig. 3. Six threads run simultaneously, utilizing the multi-core architecture.

The first thread is the driver thread, which performs basic operations, such as data acquisition and image rectification.

The dense tracking thread performs keyframe-to-frame direct dense tracking while taking the pixel disparity noise into account. The angular prior from integration of gyroscope measurements initializes the incremental rotation. This thread also identifies instantaneous tracking performance, detects tracking failure and determines whether to add a new keyframe. A disparity map is computed using the standard block matching algorithm if a new keyframe is added. The visual measurements and their corresponding frames are stored in a frame list buffer for further processing by the optimization thread (Sect. IV-C).

The optimization thread maintains a sliding window of states and measurements, and checks the frame list buffer periodically. If it is not empty, all the frames within the buffer will be added into the sliding window. If a keyframe is added, loop-closure detection is performed to find possible visual connections between keyframes. Graph-optimization is then applied to find the maximum a posteriori estimate of all the states within the sliding window using measurements from IMU pre-integration (Sect. IV-B), multi-constrained relative pose measurements (Sect. IV-D) and the prior. A two-way marginalization scheme that selectively removes states is used in order to both bound the computational complexity, IMU integration time and maximize the information stored within the sliding window (Sect. IV-F).

The latest state information after graph-based optimization is sent to the unscented Kalman filter (UKF)-based smoothing thread, which generates high-rate state estimation for quadrotor controll (Sect. IV-H).

The trajectory generation and control thread (Sect. IV-I) generates a smooth trajectory once the user specifies waypoints, maximum velocities and maximum accelerations. After that, this thread gives commands to the controller.

The controller thread receives commands from the trajectory generation and control thread and state information from the UKF-based smoothing thread. It then calculates the thrust and the desired attitude accordingly. Finally, commands are sent by the controller to the quadrotor for execution (Sect. IV-J).

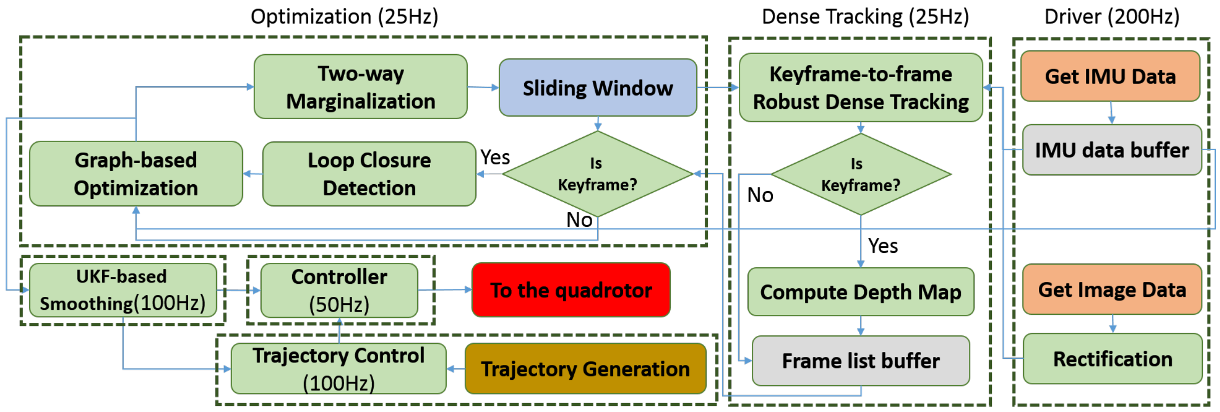


Fig. 3. The pipeline of our proposed system. It consists of six modules running in separate threads to ensure real-time availability of state estimates. The driver thread runs at 200Hz, dense tracking thread runs at 25Hz, optimization thread runs at 25Hz, UKF-based smoothing thread runs at 100Hz, trajectory control and generation thread runs at 100Hz, and controller thread runs at 50Hz.

IV. Semi-tightly-coupled Sensor Fusion

We consider $(\cdot)^k$ as the camera frame, while taking the k^{th} image, and $(\cdot)^b$ as the instantaneous IMU body frame. Without loss of generality, we assume that the cameras and the IMU are aligned. The camera-IMU sensor suite is rigidly mounted, with intrinsic and extrinsic parameters calibrated beforehand. $\mathbf{p}_Y^X$, $\mathbf{v}_Y^X$ and $\mathbf{R}_Y^X$ are the 3D position, velocity and rotation of camera frame Y with respect to frame X. We also have the corresponding quaternion ($\mathbf{q}_Y^X = [q_x, q_y, q_z, q_w]$) representation for rotation. Hamilton notation is used for quaternions.

Given two time instants that correspond to two images, we can write the IMU propagation model for position, velocity and rotation with respect to the first state of the system as in [17]:

$$\begin{aligned}
\mathbf{p}_{k+1}^0 &= \mathbf{p}_k^0 + \mathbf{R}_k^0 \mathbf{v}^k \Delta t - \mathbf{g}^0 \Delta t^2/2 + \mathbf{R}_k^0 \boldsymbol{\alpha}_{k+1}^k \\
\mathbf{v}^{k+1} &= \mathbf{R}_k^{k+1}(\mathbf{v}^k + \boldsymbol{\beta}_{k+1}^k - \mathbf{R}_0^k \mathbf{g}^0 \Delta t) \\
\mathbf{q}_{k+1}^0 &= \mathbf{q}_k^0 \otimes \mathbf{q}_{k+1}^k,
\end{aligned} \tag{1}$$

where Δt is the interval between two image acquisitions, and $\mathbf{g}^0$ is the gravity vector expressed in the first state of the system. $\boldsymbol{\alpha}_{k+1}^k$ and $\boldsymbol{\beta}_{k+1}^k$ can be obtained by integrating the IMU measurements between time instants k and $k+1$, with the definition detailed in Sect. IV-B.

A. State Estimation Formulation

We set the position and rotation of the first state to be zero. The initial velocity and gravity vector can be obtained using the online initialization method presented in [18]. The full state vector is defined as

$$\begin{aligned}
\mathcal{X} &= [\mathbf{x}_0^0, \mathbf{x}_1^0, ..., \mathbf{x}_N^0] \\
\mathbf{x}_k^0 &= [\mathbf{p}_k^0, \mathbf{v}^k, \mathbf{q}_k^0] \\
\mathbf{p}_0^0 &= [0,0,0], \ \ \mathbf{q}_k^0 = [0,0,0,1].
\end{aligned} \tag{2}$$

We aim to obtain a maximum a posteriori (MAP) estimate by minimizing the sum of the Mahalanobis norm of the weighted visual measurement residuals, inertial measurement residuals and the prior:

$$\begin{aligned}
\min_{\mathcal{X}} \ (\mathbf{b}_p - \boldsymbol{\Lambda}_p \mathcal{X}) + \sum_{k \in S_i} ||r_{S_i}(\hat{\mathbf{z}}_{k+1}^k, \mathcal{X})||^2_{\mathbf{P}_{k+1}^k} + \\
\sum_{(i,j)\in S_c} ||r_{S_c}(\hat{\mathbf{z}}_i^j, \mathcal{X})||^2_{(\mathbf{W}_i^j)^{-1}\mathbf{P}_i^j}
\end{aligned} \tag{3}$$

where $\boldsymbol{\Lambda}_p$ and $\mathbf{b}_p$ are the prior matrix and prior residual vector respectively, and S_i and S_c are the set of inertial and visual measurements respectively. $r_{S_i}(\hat{\mathbf{z}}_{k+1}^k, \mathcal{X})$ is the residual function that measures the residual between the inertial measurements and $\mathcal{X}$ with covariance $\mathbf{P}_{k+1}^k$, and $r_{S_c}(\hat{\mathbf{z}}_i^j, \mathcal{X})$ is the residual function that measures the reprojection error between visual measurements and $\mathcal{X}$ with covariance $\mathbf{P}_i^j$. Since visual measurements are subject to failure, we add a diagonal matrix $\mathbf{W}_i^j$ to weight the influence of $r_{S_c}(\hat{\mathbf{z}}_i^j, \mathcal{X})$.

Inertial measurements are obtained by IMU pre-integration (Sect. IV-B) and visual measurements are obtained by multi-constrained dense alignments (Sect. IV-D).

B. IMU Pre-integration

In this work, we adopt our proposed IMU preintegration method [8] in this work, which summarizes IMU measurements in a local frame and gets rid of the need for re-integration when the linearization point changes. The integration from IMU measurements between time instants k and $k+1$ is

$$\hat{\mathbf{z}}_{k+1}^k = \begin{bmatrix} \hat{\boldsymbol{\alpha}}_{k+1}^k \\ \hat{\boldsymbol{\beta}}_{k+1}^k \\ \hat{\mathbf{q}}_{k+1}^k \end{bmatrix} = \begin{bmatrix} \iint_{t\in[k,k+1]} \mathbf{R}_t^k \mathbf{a}_t^b dt^2 \\ \int_{t\in[k,k+1]} \mathbf{R}_t^k \mathbf{a}_t^b dt \\ \int_{t\in[k,k+1]} \frac{1}{2} \begin{bmatrix} -\lfloor \boldsymbol{\omega}^{b_t} \times \rfloor & \boldsymbol{\omega}^{b_t} \\ -\boldsymbol{\omega}^{b_t\,T} & \mathbf{0} \end{bmatrix} \mathbf{q}_t^k dt \end{bmatrix}, \tag{4}$$

where $\mathbf{a}_t^b$ and $\boldsymbol{\omega}^{b_t}$ are the instantaneous linear acceleration and angular velocity in the IMU body frame at time t respectively. The residual function between the states and the IMU measurement is defined as

$$r_{S_i}(\hat{\mathbf{z}}^k_{k+1}, \mathcal{X}) = \begin{bmatrix} \delta\boldsymbol{\alpha}^k_{k+1} \\ \delta\boldsymbol{\beta}^k_{k+1} \\ \delta\boldsymbol{\theta}^k_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{R}^k_0(\mathbf{p}^0_{k+1} - \mathbf{p}^0_k + \mathbf{g}^0\frac{\Delta t^2}{2}) - \mathbf{v}^k\Delta t - \hat{\boldsymbol{\alpha}}^k_{k+1} \\ \mathbf{R}^k_0(\mathbf{R}^0_{k+1}\mathbf{v}^{k+1} + \mathbf{g}^0\Delta t) - \mathbf{v}^k - \hat{\boldsymbol{\beta}}^k_{k+1} \\ 2[(\hat{\mathbf{q}}^k_{k+1})^{-1}(\mathbf{q}^0_k)^{-1}\mathbf{q}^0_{k+1}]_{xyz} \end{bmatrix}. \tag{5}$$

The covariance $\mathbf{P}^k_{k+1}$ can be calculated by iteratively linearizing the continuous-time dynamics of the error term and then updating it with discrete-time approximation. Detailed derivation can be found in [8].

C. Uncertainty-aware Dense Tracking

In the uncertainty-aware dense tracking module, we assume that image brightness is the same for the reference frame i and tracked frame j, and calculate the rigid-body transformation, denoted as $\mathbf{T}^j_i = \{\mathbf{t}^j_i, \mathbf{R}^j_i\} \in \mathbf{SE}(3)$, where $\mathbf{t}^j_i$ and $\mathbf{R}^j_i$ are translation and rotation respectively, that minimizes the intensity differences and takes inverse depth uncertainty into account:

$$\mathbf{T}^{j^*}_i = \arg\min_{\mathbf{T}^j_i} \iint \frac{1}{\sigma^2_{\mathbf{u}}}\delta\mathbf{I}(\mathbf{T}^j_i, \mathbf{u}, \rho_{\mathbf{u}})d\mathbf{u} \tag{6}$$

$$\delta\mathbf{I}(\mathbf{T}^j_i, \mathbf{u}, \rho_{\mathbf{u}}) = \mathbf{I}_i[\mathbf{u}] - \mathbf{I}_j[\pi(\mathbf{R}^j_i\pi^{-1}(\mathbf{u}, \rho_{\mathbf{u}}) + \mathbf{t}^j_i)], \tag{7}$$

where $\mathbf{u}$ is the coordinate of a pixel, $\rho_{\mathbf{u}}$ is the inverse depth of pixel $\mathbf{u}$, $\sigma^2_{\mathbf{u}}$ is the covariance of $\delta\mathbf{I}(\mathbf{T}^j_i, \mathbf{u}, \rho_{\mathbf{u}})$, $\mathbf{I}_k[\mathbf{u}]$ is the intensity value of pixel $\mathbf{u}$ in image k with covariance $\sigma^2_{\mathbf{I}_k}$, $\pi(\cdot) : \mathbb{R}^3 \to \mathbb{R}^2$ is the projection function that projects a 3-D point $\mathbf{f} = [x, y, z]^{\mathrm{T}}$ into the image coordinate $\mathbf{u} = (u, v)$, and $\pi^{-1}(\cdot)$ is the inverse projection function.

Since the inverse depth map is calculated using the block matching algorithm, i.e., $\rho_{\mathbf{u}} = s\lambda_{\mathbf{u}}$, where $\lambda_{\mathbf{u}}$ is the disparity of pixel $\mathbf{u}$ and s is a scaling constant related to the length of the stereo baseline and the focus length of the camera, we model the noise of $\lambda_{\mathbf{u}}$ as zero-mean white Gaussian with covariance $\sigma^2_{\lambda_{\mathbf{u}}}$. The covariance of $\rho_{\mathbf{u}}$ is

$$\sigma^2_{\rho_{\mathbf{u}}} = s^2\sigma^2_{\lambda_{\mathbf{u}}}. \tag{8}$$

We assume that the error sources are independent and approximate $\sigma^2_{\mathbf{u}}$ with first-order covariance propagation

$$\sigma^2_{\mathbf{u}} = \mathbf{J}^T_{\mathbf{u}} \begin{bmatrix} \sigma^2_{\mathbf{I}_i} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma^2_{\mathbf{I}_j} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sigma^2_{\mathbf{T}^j_i} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma^2_{\rho_{\mathbf{u}}} \end{bmatrix} \mathbf{J}_{\mathbf{u}}, \tag{9}$$

where $\mathbf{J}_{\mathbf{u}}$ is the Jacobian of $\delta\mathbf{I}(\mathbf{T}^j_i, \mathbf{u}, \rho_{\mathbf{u}})$ with respect to $\mathbf{I}_i$, $\mathbf{I}_j$, $\mathbf{T}^j_i$ and $\rho_{\mathbf{u}}$.

We adopt the Gauss-Newton approach on the Lie-manifolds to solve (6), which iteratively re-linearizes (6) around the current estimate $\hat{\mathbf{T}}^j_i$ and then performs incremental update until convergence:

$$\hat{\mathbf{T}}^j_i \leftarrow \hat{\mathbf{T}}^j_i \otimes \exp(\boldsymbol{\xi}), \tag{10}$$

where $\boldsymbol{\xi} = (\delta\mathbf{t}^j_i, \delta\boldsymbol{\theta}^j_i) \in \mathfrak{se}(3)$ is the minimum dimension error state. More details about Lie algebra $\mathfrak{se}(3)$ and Lie group $\mathbf{SE}(3)$ can be found in [19].

Augmenting (6) with linearization leads to the following linear system:

$$\mathbf{J}^{\mathrm{T}}\mathbf{W}\mathbf{J}\boldsymbol{\xi} = \mathbf{J}^{\mathrm{T}}\mathbf{W}\mathbf{r}, \tag{11}$$

where $\mathbf{J}$ is a Jacobian matrix formed by stacking Jacobians of the image intensity differences (6) with respect to $\boldsymbol{\xi}$, $\mathbf{r}$ is the corresponding intensity differences vector, and $\mathbf{W}$ is a diagonal matrix that encodes the uncertainty.

In the real implementation, we adopt image pyramids to increase the convergence region and handle large movements. Only pixels with noticeable gradients are used, for efficiency reasons. Moreover, the angular prior from integration of the IMU instantaneous angular velocity is used for initializing the incremental rotation.

The visual measurement $\hat{\mathbf{z}}^j_i$ in (3) is $\hat{\mathbf{z}}^j_i = \mathbf{T}^{j^*}_i$, and the residual function is defined as

$$r_{S_c}(\hat{\mathbf{z}}^j_i, \mathcal{X}) = \begin{bmatrix} \delta\mathbf{t}^j_i \\ \delta\boldsymbol{\theta}^j_i \end{bmatrix} = \begin{bmatrix} \mathbf{R}^j_0(\mathbf{p}^0_i - \mathbf{p}^0_j) - \hat{\mathbf{t}}^j_i \\ 2[(\hat{\mathbf{q}}^j_i)^{-1}(\mathbf{q}^0_j)^{-1}\mathbf{q}^0_i]_{xyz} \end{bmatrix}, \tag{12}$$

where $\hat{\mathbf{q}}^j_i$ is the quaternion representation of $\hat{\mathbf{R}}^j_i$. It can be derived mathematically that the corresponding covariance $\mathbf{P}^j_i$ is the inverse of the Hessian matrix $\mathbf{J}^{\mathrm{T}}\mathbf{W}\mathbf{J}$ at the final iteration.

D. Multi-constrained Dense Alignments

Uncertainty-aware dense tracking is performed between the latest keyframe within the sliding window and the incoming frame once a new frame comes. If the tracking succeeds, a visual measurement and its corresponding covariance are inserted into the sliding window.

In addition, since significant drift may occur after aggressive motions, we introduce a local loop-closure module for recovery. Once a new keyframe is added, loop-closure detection is performed to seek possible visual measurements between existing keyframes within the sliding window and the new keyframe using uncertainty-aware dense tracking. Note that a cross check is adopted to avoid incorrect loop closure. If the two corresponding estimated rigid-body transformations are consistent, the cross check is successfully passed.

E. Dense Tracking Failure Detection

Although our proposed uncertainty-aware dense tracking takes the inverse depth uncertainty into account and increases the robustness compared to the traditional methods, it still fails in extreme cases, such as during aggressive motions within textureless surroundings. Detection of dense tracking failure is of vital importance to our system. As shown in Sect. IV-C, covariance matrix $(\mathbf{J}^{\mathrm{T}}\mathbf{W}\mathbf{J})^{-1}$ tells us about the dense tracking performance. We consider the dense tracking to be a failure if the covariance is greater than a certain threshold. Also, since IMU measurements are reliable in the short-term, dense tracking is considered to have failed if its incremental transformation estimation is not consistent with the prior from the IMU integration.

F. Two-way Marginalization

A two-way marginalization scheme [18] is used to maintain a sliding window of states and convert measurements corresponding to the marginalized states into a prior. Firstly, since the memory and computational resources of our system are limited, we can only optimize a certain number of states and visual or inertial measurements for real-time performance setting. Secondly, the effectiveness of multi-constrained dense alignment and drift elimination depend on whether an older state is kept within the sliding window. Thirdly, we ensure that the time interval for each IMU preintegration is bounded in order to bound the accumulated error. By two-way marginalization, all information of the removed states is kept and computation complexity is bounded, which is fundamentally different from traditional keyframe-based approaches that simply drop non-keyframes. Front marginalization removes the second newest frame, while back marginalization removes the oldest keyframes within the sliding window. In this work, to preserve all the information (visual and inertial measurements) related to non-keyframes, we only perform front marginalization after the newest state comes.

Note that the criterion to select whether to use front or back marginalization is based on the result from the dense tracking failure detection module introduced in Sect. IV-E. The second newest state will be marginalized in the next round if the dense tracking is good and the second newest state is near to the current keyframe. Otherwise, the oldest state will be marginalized. The distance is thresholded by a weighted combination of translation and rotation between the latest keyframe and the second newest frame.

G. Optimization with Robust Norm

Based on the residual functions defined in (5) and (12), we operate on the error state and optimize (3) using the Gaussian-Newton method, which iteratively minimizes:

$$\min_{\delta\mathcal{X}} \; (\mathbf{b}_p - \mathbf{\Lambda}_p\mathcal{X}) + \sum_{k\in S_i} ||r_{S_i}(\hat{\mathbf{z}}^k_{k+1}, \mathcal{X}) + \mathbf{H}^k_{k+1}\delta\mathcal{X}||^2_{\mathbf{P}^k_{k+1}} \tag{13}$$

$$+ \sum_{(i,j)\in S_c} ||r_{S_c}(\hat{\mathbf{z}}^j_i, \mathcal{X}) + \mathbf{H}^j_i\delta\mathcal{X}||^2_{(\mathbf{W}^j_i)^{-1}\mathbf{P}^j_i}$$

and then updates

$$\hat{\mathcal{X}} = \hat{\mathcal{X}} \oplus \delta\mathcal{X} \tag{14}$$

until convergence. $\mathbf{H}^k_{k+1}$ and $\mathbf{H}^j_i$ are the Jacobian matrices of inertial measurements and visual measurements with respect to the states.

To increase the robustness of our proposed system, $\mathbf{W}^j_i$ changes in each iteration to further eliminate possible outliers in dense tracking that pass the failure detection (Sect. IV-E). $\mathbf{W}^j_i$ is computed according to the Huber thresholding on the current estimate:

$$(\mathbf{W}^j_i)_{ul} = \begin{cases} \mathbb{I}_{3\times3}, & \text{if } ||\mathbf{R}^j_0(\mathbf{p}^0_j - \mathbf{p}^0_i) - \hat{\mathbf{t}}^j_i|| \le c_t \\ \frac{c_t}{||\mathbf{R}^j_0(\mathbf{p}^0_j - \mathbf{p}^0_i) - \hat{\mathbf{t}}^j_i||}\mathbb{I}_{3\times3}, & \text{otherwise} \end{cases} \tag{15}$$

$$(\mathbf{W}^j_i)_{lr} = \begin{cases} \mathbb{I}_{3\times3}, & \text{if } ||2[(\hat{\mathbf{q}}^j_i)^{-1}(\mathbf{q}^0_j)^{-1}\mathbf{q}^0_i]_{xyz}|| \le c_a \\ \frac{c_a}{||2[(\hat{\mathbf{q}}^j_i)^{-1}(\mathbf{q}^0_j)^{-1}\mathbf{q}^0_i]_{xyz}||}\mathbb{I}_{3\times3}, & \text{otherwise} \end{cases} \tag{16}$$

where $(\mathbf{W}^j_i)_{ul}$ is the upper left 3×3 matrix of $\mathbf{W}^j_i$, $(\mathbf{W}^j_i)_{lr}$ is the lower right 3×3 matrix of $\mathbf{W}^j_i$, $\mathbb{I}_{3\times3}$ is an identity matrix, and c_t and c_a are the given translation and angular threshold respectively.

H. UKF-based Smoothing

The 25Hz state estimation from the optimization module alone is not sufficient to control the quadrotor. The flight controller with a separate IMU streams out measurements at 100 Hz. As a result, we employ an UKF to smooth poses and velocities at 100 Hz [20]. The inputs to the UKF-based smoothing thread are absolute poses and velocities with respect to the zero frame from the optimization module and the IMU measurements from the flight controller. The states in the UKF are $\mathbf{x} = [\mathbf{r}, \dot{\mathbf{r}}, \mathbf{\Psi}, \mathbf{b}_a]^{\mathrm{T}}$, where $\mathbf{\Psi} = [\phi, \theta, \psi]^{\mathrm{T}}$ represents the roll, pitch, and yaw angles of the quadrotor respectively, $\mathbf{b}_a = [b_{a_x}, b_{a_y}, b_{a_z}]^{\mathrm{T}}$ is the accelerometer's bias expressed in the body frame, and $\mathbf{r}$ and $\dot{\mathbf{r}}$ are the positions and velocities of the quadrotor respectively.

I. Trajectory Generation

Given a set of expected way-points and the maximum velocity and acceleration, we would like to generate a smooth trajectory so that the quadrotor can follow it while moving as fast as possible. We utilize a polynomial trajectory generation algorithm [21] that runs onboard.

J. Control

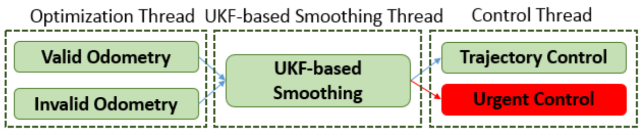


Fig. 4. Control pipeline.

The controller aims to control the quadrotor to follow the generated trajectory. Suppose the difference between the current and expected (denoted with *) position and velocity is $\mathbf{e}_p = \mathbf{r}_t - \mathbf{r}^*_t$, $\mathbf{e}_v = \dot{\mathbf{r}}_t - \dot{\mathbf{r}}^*_t$. The force vector for the controller is:

$$\mathbf{F} = -K_p\mathbf{e}_p - K_v\mathbf{e}_v + mg\mathbf{z}_W + m\ddot{\mathbf{r}}^*, \tag{17}$$

where K_p and K_v are the constants, and $\mathbf{z}_W$ is the vertical axis in the world [22]. The state information from the UKF is used as the feedback of the controller. Fig. 4 shows the pipeline of the control actions. The outputs from the optimization thread are vaild odometry (tracking is good), keyframe odometry (new keyframe inserted) and invalid odometry (tracking is bad, merely integration from inertial measurements is available). Normally, trajectory control takes charge. However, if invalid odometry lasts more than two seconds, urgent control that adjusts the roll and pitch angle to be zero, yaw angle to be fixed and position height to be fixed, will be enabled for safety reason.

V. Experiments

The experiment platform is the Matrice 100 quadrotor from DJI[1] with an SDK for execution of control commands (Fig. 1). Our visual-inertial sensor suite equipped in the platform is the VI-Sensor[2], which consists of a MEMS IMU and two global shutter cameras with a fronto-parallel stereo configuration. A high-level computer, NUC from Intel, with a low-power dual-core CPU i5-4250U running at 1.3GHz and 16GB RAM is used for computing in our proposed system. The operating system installed on Intel NUC is Ubuntu 14.04. The mass of our platform is 3210g. All the algorithms are developed in C++, with ROS as the interfacing robotics middleware. The frequencies of the IMU data and stereo camera data are 200 Hz and 25 Hz respectively. The cameras have factory pre-calibrated intrinsics and extrinsics.

Component	Average Computation Time	Thread
Driver	1ms	0
Dense Tracking	13ms	1
Block Matching	8ms	1
Graph Optimization	6ms	2
Two-way Marginalization	3ms	2
Loop Closure	30ms	2
UKF-based Smoothing	1ms	3
Controller	1ms	4
Trajectory Generation	3ms	5
Trajectory Control	1ms	5

TABLE I
AVERAGE COMPUTATION TIME OF MAIN TIME-CONSUMING COMPONENTS OF OUR PROPOSED SYSTEM.

A. Real-time Implementation

For the benefit of real-time performance, we set the finest resolution for uncertainty-aware dense tracking to be 320×240 and the number of levels of the pyramid to be 3. Our implementation of dense tracking is built on top of the open-source and vectorized implementation of [3]. The noticeable gradient threshold of the uncertainty-aware dense tracking is 5 and the sliding window size is 30. The block matching algorithm we use is the simplest and fastest one in OpenCV (Stereo BM). We empirically find that, though the disparity map is not satisfied using this native block matching algorithm, the performance of our tracking module is quite good since it models and approximates the source of noise well. The computing time of each component is summarized in Table I.

[1]https://developer.dji.com/matrice-100/
[2]http://www.skybotix.com/

The way to communicate between threads in our system is memory share. Calculation is the main delay of our system. Our system performance can be guaranteed if the time delay caused by calculation is limited.

B. Performance with Normal Speed Flights

In this experiment, we control the quadrotor to fly at a speed about 1 m/s and compare the estimation accuracy of our system with a keypoint-based method (fovis [10]) and semi-dense tracking [12]. Since the method in [12] is for monocular vision, we modify it by initializing the depth values using stereo block matching when a new keyframe is inserted instead of generating random ones or using continuous depth fusion for fair comparison. The total flying distance is about 80 meters, as shown in Fig. 5. We find that there is occasionally tracking loss with fovis, and therefore the final drift is large (thus we do not plot it). The final position drifts of semi-dense tracking and our system are 0.45m and 0.23m respectively. Since the flying speed is not fast, both semi-dense tracking and our approach works well and have good performance. Moreover, our proposed system fully utilizes the advance of multi-constrained relative pose measurements and semi-tightly-coupled optimization, resulting in less drift.

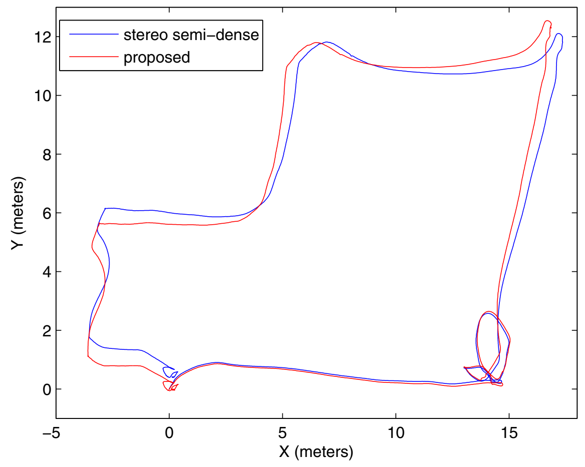


Fig. 5. We compare the estimation accuracy of our proposed system with semi-dense tracking [12] of quadrotor flights at normal speed.

C. Performance with Aggressive Flights

We demonstrate the comprehensive performance of the proposed system with aggressive motions by generating expected trajectories (figure-eight and cross pattern) and having the quadrotor follow them using feedback control from *onboard state estimations*. The reason we adopt the figure-eight and cross pattern is that the quadrotor will have large linear and angular changes that lead to aggressive motions when following these two trajectories. The robustness and smoothness of onboard motion estimation are of vital importance to this experiment. Snapshots of these flights are shown in Fig. 2. We compare the real-time estimation accuracy and robustness between our system and state-of-the-art methods, such as the keypoint-based method fovis [10], semi-dense tracking [12] (adapted to be stereo the same as in the last experiment) and the loosely-coupled dense VINS dense-EKF [16], and show that our system is the only one that is able to handle these challenging cases. Ground truth data obtained by the OptiTrack Prime 41 system is provided for comparison.

The trajectories of the ground truth and those from the different approaches are shown in Fig. 6.(a) and Fig. 7.(a). The statistics of angular rates, linear velocities and linear accelerations are shown in Fig. 6.(b) and Fig. 7.(b). The angular rates are measured by IMU. Linear velocities and accelerations are calculated by our proposed system. It can be easily seen from the trajectories that both the methods in [10] and [12] work poorly during aggressive flight. We give a detailed visual comparison of angles and positions between our system, dense-EKF and the ground truth in Fig. 6.(c) and (d) and Fig. 7.(c) and (d). While the performance of the method [16] is good with the cross pattern, it is bad with the figure-eight. Our estimator is the best in terms of both accuracy and robustness with both trajectories and the final statistics are summarized in Table II. Our method is the only approach with observability of roll/pitch and multi-constrained visual measurements, resulting in least drift. Also, the semi-tightly-coupled fusion scheme of our system provides accurate and smooth trajectories that benefit the controller.[3]

Trajectory	figure-eight	cross pattern
Maximum Angular Rate (degree/s)	183.9	245.1
Maximum Linear Velocity (m/s)	2.8	4.2
Maximum Linear Acceleration (m/s^2)	6.7	9.6
Position-x Drift (m)	0.09	0.07
Position-y Drift (m)	0.03	0.03
Position-z Drift (m)	0.24	0.09
Yaw Drift (degree)	3.07	0.37

TABLE II

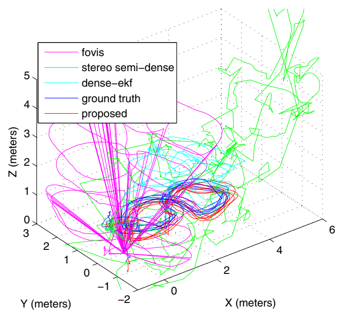
SUMMARY OF OUR SYSTEM COMPARED TO GROUND TRUTH DATA.

Note that there are limits on maximum attitude with the SDK of the M100. We will test our approach with more aggressive motions in the future if the platform hardware improves. More experiments and experiment details can be found in the accompanying video mentioned above.

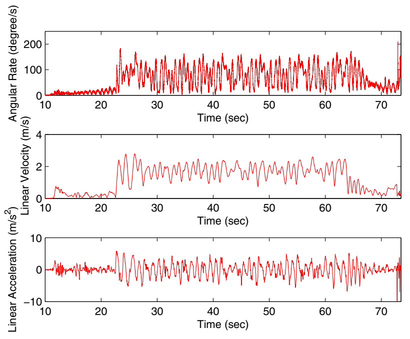
VI. CONCLUSIONS AND FUTURE WORK

We propose a real-time, robust and fully integrated system for quadrotors to fly with aggressive motions. The core of our system is a semi-tightly-coupled and probabilistic sensor fusion scheme that provides robust state estimation. Online experiments have verified the superior performance of our system. In the future, we will integrate obstacle avoidance modules into our system.

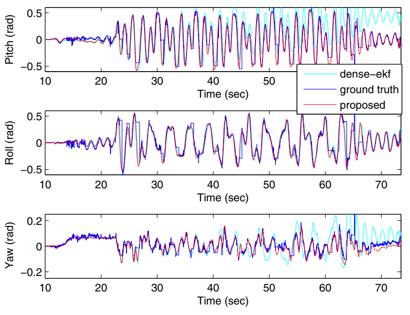
[3]Since the ground truth data is not good due to limited space and hardware, we do not include the standard derivation of linear velocities and roll/pitch in Table II.



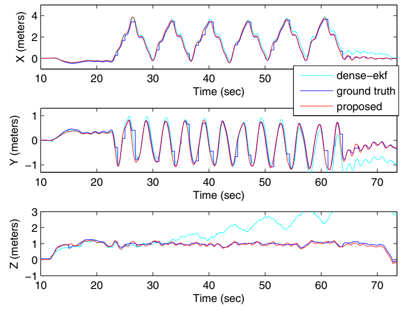
(a) Trajectories



(b) Statistics

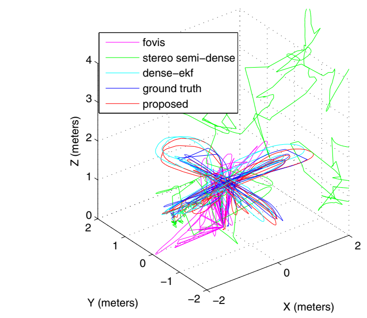


(c) Angles

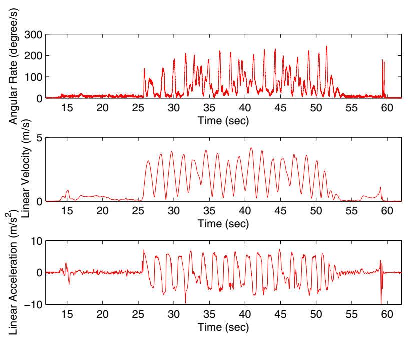


(d) Positions

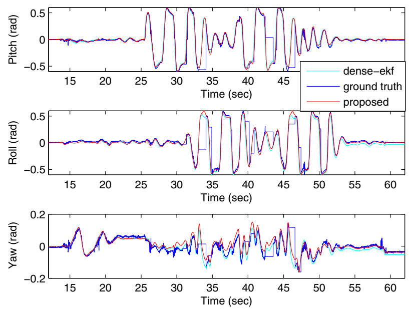
Fig. 6. Performance of our proposed method compared with ground truth obtained by OptiTrack, fovis [10], semi-dense tracking [12] and dense-EKF [16]. (a) Estimated trajectory of figure-eight. (b) Statistics of angular rates, linear velocities and linear accelerations during the flight. (c) Comparison of angles. (d) Comparison of positions. Ground truth data is not good due to limited space and hardware.



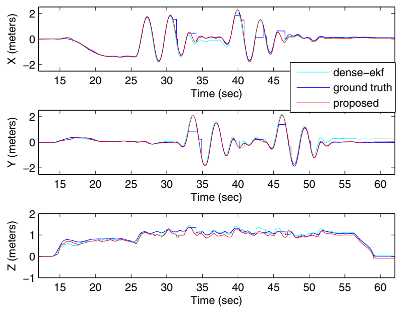
(a) Trajectories



(b) Statistics



(c) Angles



(d) Positions

Fig. 7. Performance of our proposed method compared with ground truth obtained by OptiTrack, fovis [10], semi-dense tracking [12] and dense-EKF [16]. (a) Estimated trajectory of the cross pattern. (b) Statistics of angular rates, linear velocities and linear accelerations during the flight. (c) Comparison of angles. (d) Comparison of positions. Ground truth data is not good due to limited space and hardware.

References

[1] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, “Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor,” in *Proc. of Robot.: Sci. and Syst.*, Berlin, Germany, 2013.

[2] R. A. Newcombe, S. Lovegrove, and A. J. Davison, “DTAM: dense tracking and mapping in real-time,” in *IEEE International Conference on Computer Vision*, 2011, pp. 2320–2327.

[3] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” in *European Conference on Computer Vision (ECCV)*, September 2014.

[4] Y. Ling and S. Shen, “Dense visual-inertial odometry for tracking of aggressive motions,” in *Proc. of the IEEE Intl. Conf. on Robot. and Bio.*, 2015.

[5] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, “Consistency analysis and improvement of vision-aided inertial navigation,” *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 158–176, Feb. 2014.

[6] M. Li and A. Mourikis, “High-precision, consistent EKF-based visual-inertial odometry,” *Intl. J. Robot. Research*, vol. 32, no. 6, pp. 690–711, May 2013.

[7] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Fraundorfer, E. Kosmatopoulos, A. Martinelli, M. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss, “Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments,” *IEEE Robot. Autom. Mag.*, vol. 21, no. 3, 2014.

[8] S. Shen, N. Michael, and V. Kumar, “Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs,” in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Seattle, WA, May 2015.

[9] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, “Keyframe-based visual-inertial SLAM using nonlinear optimization,” in *Proc. of Robot.: Sci. and Syst.*, Berlin, Germany, June 2013.

[10] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, “Visual odometry and mapping for autonomous flight using an RGB-D camera,” in *Proc. of the Intl. Sym. of Robot. Research*, Flagstaff, AZ, Aug. 2011.

[11] C. Kerl, J. Sturm, and D. Cremers, “Robust odometry estimation for rgb-d cameras,” in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, May 2013.

[12] J. Engel, J. Sturm, and D. Cremers, “Semi-dense visual odometry for a monocular camera,” in *Proc. of the IEEE Intl. Conf. Comput. Vis.*, Sydney, Australia, December 2013.

[13] C. Kerl, J. Sturm, and D. Cremers, “Dense visual slam for rgb-d cameras,” in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2013.

[14] M. K. G. Huang and J. Leonard, “Towards consistent visual-inertial navigation,” in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Hong Kong, June 2014.

[15] A. I. Comport, E. Malis, and P. Rives, “Real-time quadrifocal visual odometry,” in *Intl. J. Robot. Research*, 2010.

[16] S. Omari, M. Bloesch, P. Gohl, and R. Siegwart, “Dense visual-inertial navigation system for mobile robots,” in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2015.

[17] T. Lupton and S. Sukkarieh, “Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions,” *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 61–76, Feb. 2012.

[18] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, “Initialization-free monocular visual-inertial estimation with application to autonomous MAVs,” in *Proc. of the Intl. Sym. on Exp. Robot.*, Morocco, 2014.

[19] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-d vision: from images to geometric models*. Springer Science Business Media, 2012, vol. 26.

[20] R. Van Der Merwe, E. A. Wan, S. Julier, *et al.*, “Sigma-point kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation,” in *Proc. of AIAA Guidance, Navigation and Control Conference*, 2004.

[21] N. R. Charles Richter, Adam Bry, “Polynomial trajectory planning for quadrotor flight,” in *Workshop on the Intl. Conf. on Field and Service Robot.*, 2013.

[22] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, May 2011.